Journal of Computational and Applied Mathematics 35 (1991) 291–296
North-Holland

Polynomial algorithms to finite Veber problem for a tree network

Anatoly V. Panyukov and Boris V. Pelzwerger
Informatics Department, Chelyabinsk State Technical University, 76 Lenin ave., Chelyabinsk, 454080 USSR

Received 2 August 1990

Abstract

Panyukov, A.V. and B.V. Pelzwerger, Polynomial algorithms to finite Veber problem for a tree network, Journal of Computational and Applied Mathematics 35 (1991) 291–296.

A polynomial algorithm to a finite Veber problem for a tree network is presented. Space- and time-complexity estimations of this algorithm are given. The space complexity of this algorithm may be decreased for problems with an algorithmic representation of some input data. Examples of such problems and algorithms to them are given.

Keywords: Algorithm, complexity, extreme problem, graph, tree.

1. Introduction

A finite Veber problem is an extreme problem $\Theta(G, V, b, c)$:

$$f(\phi) = \sum_{j \in J} c(j, \phi(j)) + \sum_{[i,j] \in E} b([i, j], \phi(i), \phi(j)) + \min_{\phi: J \to V} \tag{1}$$

for a given graph $G = (J, E)$, a finite set V, a symmetrical mapping $b: E \times V^2 \to \mathbb{R}$ and a mapping $c: J \times V \to \mathbb{R}$.

Problem (1) represents a weakened version of a square assigned problem which does not demand the mapping ϕ to be an injection. The polynomial algorithms to solve (1) are known provided the mappings b and c are special [3,5]. These notes present the polynomial algorithm to solve (1) provided the graph G is a tree and mappings b and c are arbitrary. Note that the problem will be NP-hard if the mapping ϕ is an injection [4].

2. General case

Let $G = (J, E)$ be a tree, N be a cardinality of the set J, i.e., $N = |J|$. Let a hanging node ω be received as a root of the tree G that induces the partition order relation

$$P = \{(u, v) \mid u, v \in J, v \text{ lies on the single path between } u \text{ and } w\}.$$

```
Algorithm LAYOUT (input: N, F(*), c(*, *), b(*, *, *), V; output: φ(*))
begin
  for i := 1 up to N − 1 do
    for each v ∈ V do
      begin
        c(F(i), v) := c(F(i), v) + min_{u∈V}(c(i, u) + b([i, F(i)], u, v));
        A(i, v) := arg min_{u∈V}(c(i, u) + b([i, F(i)], u, v));
      end
  φ(N) := arg min_{u∈V}(c(N, u));
  for i := n − 1 down to 1 do
    φ(i) := A(i, φ(F(i)));
end
```

Fig. 1.

Further it is supposed that the nodes of the set $J = \{j_n\}_{n=1}^{N}$ are numbered so that $(l, m) \in P \Rightarrow l < m$. The direct predecessor of a node l is denoted as $F(l)$, i.e., $F(l) = m$: $[l, m] \in E$, $l < m$.

The pidgin algol algorithm for the considered problem is represented in Fig. 1. The algorithm uses an auxiliary array $A(i, v)$, $i = 1, 2, \ldots, n$, $v \in V$. The algorithm changes the values of array c, which is a side effect.

Theorem 1. *The algorithm LAYOUT, represented in Fig. 1, solves the problem* $\Theta(G, V, b, c)$ *where G is the N-nodes numbered root tree*

$$\left\{J = \{k\}_{k=1}^{N},\ E = \{[k, F(k)]\}_{k=1}^{n-1}\right\}$$

with a direct predecessor function $F(*)$. *Its space complexity is* $O(|J||V|^2)$. *Its time complexity is* $O(|J||V|^2)$.

Proof. From the space-complexity estimation being $O(|J||V|^2)$, it follows that $O(|J||V|^2)$ memory cells are required for an input/output data, $O(|J||V|)$ memory cells for the array $A(*, *)$ and $O(1)$ ones for the rest. From the time-complexity estimation being $O(|J||V|^2)$, it follows that $O(|V|)$ operations are required to calculate $A(i, v)$ and $c(F(t), v)$ for each $i \in J$, $v \in V$ and $O(|J| + |V|)$ operations for the rest.

Further it is supposed that $G(K)$ is the subgraph of the graph G in the node set $K \subset J$, $\varphi(*)|_X$ is a restriction to X of the map φ. Let us consider the sequence of the problems $\{\Theta_n(G_n, V, b, c_n)\}_{n=1}^{N}$ in which

$$G_1 = G, \qquad c_1 = c, \tag{2}$$

and problems Θ_{n+1}, $n = 1, 2, \ldots, n-1$, are derived in the following way:

$$G_{n+1} = (J_{n+1}, E_{n+1}) = G_n(J_{n+1}), \quad J_{n+1} = J_n \setminus \{j_n\}, \tag{3a}$$

$$(\forall j \in J_{n+1} \setminus \{F(j_n)\}, \forall v \in V)\,(c_{n+1}(j, v) = c_n(j, v)), \tag{3b}$$

$$(\forall v \in V)\left(C_{n+1}(F(j_n), v) = c_n(F(j_n), v) + \min_{k \in V}\left(c_n(j_n, k) + b([j_n, F(j_n)], k, v)\right)\right). \tag{3c}$$

To finish the proof of Theorem 1, we first proof the next theorem.

Theorem 2. *There are decisions* ϕ_{Θ_n} *of problems* Θ_n, $n = 1, 2, \ldots, N$, *so that*

$$\phi_{\Theta_n} = \phi_{\Theta} \mid_{J_n}, \tag{4}$$

$$f_{\Theta_n}(\phi_{\Theta_n}) = f_{\Theta_{n+1}}(\phi_{\Theta_{n+1}}), \quad n = 1, 2, \ldots, N-1, \tag{5}$$

$$\phi_{\Theta}(j_N) = \arg \min_{k \in V} c_N(j_N, k), \tag{6}$$

$$\phi_{\Theta}(j_n) = \arg \min_{k \in V} \big(c_n(j_n, k) + b\big([j_n, F(j_n)], k, \phi_{\Theta}(F(j_n))\big)\big), \quad n = 1, 2, \ldots, N-1. \tag{7}$$

Proof. Conditions (2) and (3) give

$$\begin{aligned} f_{\Theta_n}(\phi_{\Theta_n}) &= \sum_{j \in J_n} c_n\big(j, \phi_{\Theta_n}(j)\big) + \sum_{[l,k] \in E_n} b\big([l, k], \phi_{\Theta_n}(l), \phi_{\Theta_n}(k)\big) \\ &= \sum_{j \in J_{n+1} \setminus \{F(j_n)\}} c_n\big(j, \phi_{\Theta_n}(j)\big) + \sum_{[l,k] \in E_{n+1}} b\big([l, k], \phi_{\Theta_n}(l), \phi_{\Theta_n}(k)\big) \\ &\quad + c_n\big(j_n, \phi_{\Theta_n}(j_n)\big) + c_n\big(F(j_n), \phi_{\Theta_n}(F(j_n))\big) \\ &\quad + b\big([j_n, F(j_n)], \phi_{\Theta_n}(j_n), \phi_{\Theta_n}(F(j_n))\big) \\ &\geqslant f_{\Theta_{n+1}}(\phi_{\Theta_n} \mid_{J_{n+1}}) \geqslant f_{\Theta_{n+1}}(\phi_{\Theta_{n+1}}), \quad n = 1, 2, \ldots, N. \end{aligned} \tag{8}$$

To proof the reverse inequality we construct $\bar{\phi}_{\Theta_n}$, so that

$$\bar{\phi}_{\Theta_n} \mid_{J_{n+1}} = \phi_{\Theta_{n+1}},$$

$$\bar{\phi}_{\Theta_n}(j_n) = \arg \min_{k \in V} \big(c_n(j_n, k) + b\big([j_n, F(j_n)], k, \phi_{\Theta_{n+1}}(F(j_n))\big)\big). \tag{9}$$

The equalities (2), (3) and (9) imply

$$f_{\Theta_{n+1}}(\phi_{\Theta_{n+1}}) = f_{\Theta_n}(\bar{\phi}_{\Theta_n}) \geqslant f_{\Theta_n}(\phi_{\Theta_n}). \tag{10}$$

From the last inequality it follows that $\bar{\phi}_{\Theta_n}$ may not be an optimal solution of the problem Θ_n. Comparing (8) and (9) implies (5). Moreover,

$$f_{\Theta_n}(\bar{\phi}_{\Theta_n}) = f_{\Theta_n}(\phi_{\Theta_n}) = f_{\Theta}(\phi_{\Theta}), \quad n = 1, 2, \ldots, N, \tag{11}$$

and we may consider

$$\phi_{\Theta_n} = \bar{\phi}_{\Theta_n}, \quad n = 1, 2, \ldots, N. \tag{12}$$

Equality (11) and the construction of ϕ_{Θ_n} imply (4). Problem Θ_n has the form

$$c_N(j_N, k) \to \min_{k \in V},$$

and we get (6). Finally, (7) follows from (12) and (9). □ (Theorem 2)

The algorithm execution consists of two main loops. At the first main loop the algorithm constructs the above-showed sequence of the problems $\{\Theta_n\}_{n=1}^{N}$ and calculates

$$A(i, v) = \arg \min_{u \in V} \big(c_n(i, u) + b\big([i, F(i)], u, v\big)\big), \quad n = 1, 2, \ldots, N-1, \ v \in V.$$

At the second stage the algorithm finds the meanings

$$\phi_\Theta(j_N),\ \phi_\Theta(j_{N-1}),\ldots,\phi_\Theta(j_1)$$

are found in accordance with Theorem 2. □ (Theorem 1)

Cardinally the memory capacity is used for store of the mapping b. For a number of applied network problems the algorithm space complexity may be decreased due to an algorithmic representation of the mapping b. Further the example of such a problem and an algorithm to solve one are considered.

3. Case of an algorithmic representation of the mapping b

Let a graph (V, W) of elementary communications and a mapping $d: E \times W \to \mathbb{R}^+$ be given. Further a special case of (1) when $b([i, j], u, v), [i, j] \in E$, are longs of the shortest path between $u, v \in V$ in the graph (V, W) with the edges $(l, k) \in W$ with longs $d([i, j], [l, k])$, is considered.

In a worse case the input data memory capacity of such problems are $O(|W||J|)$. But direct use of the algorithm LAYOUT proposes that all $b([i, j], u, v), [i, j] \in E, u, v \in V$, are calculated. These calculation require $|E|$ times to calculate a short distance matrix between all pairs of nodes of the graph (V, W). For an applied problem this is hard enough. The known

```
Algorithm LAYOUT (input: N, F(*), c(*, *), (V, W), d(*, *); output : φ(*))

begin
  for i := 1 up to N − 1 do
    begin
      for each v ∈ V do A(i, v) := v;
      U := φ;
      while U ≠ V do
        begin
          u := arg min_{v∈V} c(i, v);
          U := U ∪ {u};
          for each v ∈ {x | x ∈ V\U, (x, u) ∈ W} do
            begin
              h := c(i, u) + d([i, F(i)], [u, v]);
              if c(i, v) > h then
                begin
                  c(i, v) := h;
                  A(i, v) := u;
                end
            end
        end
        for each v ∈ V do c(F(i), v) := c(F(t), v) + c(i, v);
    end
  φ(N) := arg min_{u∈V} c(N, u);
  for i := N − 1 down to 1 do
    φ(i) := A(i, φ(F(i)));
end
```

Fig. 2.

Floyd–Warshall algorithm [2,6], for example, requires $O(|V|^2)$ memory cells and $O(|V|^3)$ operations.

The pidgin algol algorithm to solve such problems which is not requiring to calculate all values $b(*, *, *)$ is represented in Fig. 2.

Theorem 3. *The algorithm LAYOUT, represented in Fig. 2, solves the problem* $\Theta(G, V, b, c)$ *where G is the N-nodes numbered root tree*

$$\left(J = \{k\}_{k=1}^{N},\ E = \{[k, F(k)]\}_{k=1}^{n-1}\right)$$

with a direct predecessor function $F(*)$; $b([i, j], u, v), [i, j] \in E, u, v \in V$, *are longs of the shortest path between u and v in the graph* (V, W) *with the edges* $[l, k] \in W$ *with longs* $d([i, j], [l, k])$. *Its space complexity is* $O(|J||V|^2)$. *Its time complexity is* $O(|J||V|^2)$.

Proof. Let us consider the ith execution of the first main loop body of the algorithm. If we add the fictitious node v_0 and edges $[v_0, v]$, $v \in V$, which have the longs $c(t, v)$ to the graph (V, W), then values

$$c(i, v) = \min_{u \in V}\left(c(i, u) + b([i, F(i)], u, v)\right), \quad v \in V,$$

will be equal to longs of the shortest paths between v_0 and v in the received graph and

$$A(i, v) = \arg\min_{u \in V}\left(c(i, u) + b([i, F(i)], u, v)\right), \quad v \in V,$$

will be adjacent to v_0 nodes of this shortest path.

The problem to find these values is known as the shortest paths tree problem. The first main loop body of the algorithm LAYOUT1 is the known Dijkstra algorithm [1] to solve this problem. Its time complexity is $O(|V|^2)$, which implies the time-complexity estimation of the algorithm LAYOUT1 (Fig. 2).

The algorithm LAYOUT1 input/output data memory capacity is $O(|W||J|)$, the array $A(*, *)$ memory capacity is $O(|V||J|)$, and the memory capacity of the rest is $O(1)$, which implies the space-complexity estimation of the algorithm LAYOUT1. □

4. Conclusion

Again we may decrease the space complexity by using an algorithmic representation of the edges $[l, k] \in W$ with longs $d([i, j], [l, k]), [i, j] \in E$.

For more applied network problems

$$d([i, j], [l, k]) = f[l, k] + h[l, k] \cdot x[i, j],$$

where mappings $f: W \to \mathbb{R}^+$, $h: W \to \mathbb{R}^+$ and $x: E \to \mathbb{R}^+$ are given. It is clear in this case that the algorithm space complexity may be $O(|J||V| + |W|)$.

A further decreasing space and time complexity of the algorithm are possible by making use of the special kind of the graph (V, W).

The worked out algorithms and their computer codes are used for synthesis and development control of oil deposit transportation and communication networks.

References

[1] E.W. Dijkstra, A note on two problems in connection with graphs, *Numer. Math.* **1** (1959) 269–271.
[2] R.W. Floyd, Algorithm 97: Shortest path, *Comm. ACM* **5** (1962) 345.
[3] R.L. Francis and J.A. White, *Facilities Layout and Location: an Analytical Approach* (Prentice-Hall, Englewood Cliffs, NJ, 1974).
[4] J.-C. Picard and M. Queyranne, On the one-dimension space allocation problem, *Oper. Res.* **29** (1981) 371–391.
[5] V.A. Trubin, An effective algorithm to Veber problem with a rectangular metric, *Kibernetika* **6** (1978) 67–70 (in Russian).
[6] S.A. Warshall, Theorem on Boolean matrices, *J. Assoc. Comput. Mach.* **9** (1962) 11–12.